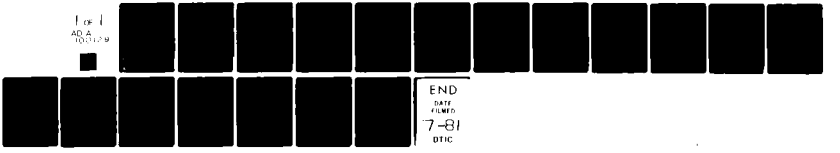
AD-A100 129 MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE F/G 9/2
DESIGN OF THE VAX INSTRUMENTATION PACKAGE (VIP).(U)
OCT 80 L W DOWDY, W S FREEZE, B G LABAW AFOSR-78-3654
UNCLASSIFIED TR-952 AFOSR-TR-81-0481 NL



END
DATE FILMED
7-81
DTIC

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER: AFOSR-TR-81-0481 | 2. GOVT ACCESSION NO.: AD-A100129 |
| 3. RECIPIENT'S CATALOG NUMBER | |
| 4. TITLE (and Subtitle): DESIGN OF THE VAX INSTRUMENTATION PACKAGE (VIP). | 5. TYPE OF REPORT & PERIOD COVERED: INTERIM |
| | 6. PERFORMING ORG. REPORT NUMBER: TR-952 |
| 7. AUTHOR(s): L.W. Dowdy, W.S. Freeze, B.G. Labaw and J.N. Reed | 8. CONTRACT OR GRANT NUMBER(s): AFOSR-78-3654 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Computer Science, University of Maryland, College Park MD 20742 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: PE61102F, 2304/A2 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB DC 20332 | 12. REPORT DATE: OCTOBER 1980 |
| | 13. NUMBER OF PAGES: 16 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | 15. SECURITY CLASS. (of this report): UNCLASSIFIED |
| | 15a. DECLASSIFICATION DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DTIC
JUN 1 1981
E

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the design of the VAX Instrumentation Package (VIP). The target machine initially is the VAX 11/780 running under the Unix operating system at the Computer Science Department of the University of Maryland.

VIP is a software monitor of VAX performance. It is designed as an aid for system monitoring, system tuning, and system modeling. For example, VIP monitors the request rate of system files. If the request rate of a particular file
(CONT.)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONT.:

becomes high, the system may be "tuned" by making the file resident in main memory. Specific parameters of the file (e.g., the average number of words transferred per file request) may be used in a queueing network model of the system to predict the performance impact of making the file resident in main memory.

A major motivation for VIP is collecting parametric values for, and providing validation of, system models of the VAX. In the past few years, sophisticated and powerful modeling techniques have been developed. A major problem in the application of these techniques is that actual systems do not monitor those input parameters that are necessary for the application of the modeling techniques. Furthermore, these modeling techniques typically provide performance measures which are more extensive than are usually measured. Validation, as well as the construction, of accurate models is thus a difficult problem.

During the design process of VIP, several existing software monitors were examined in detail. VIP includes, extends, and deletes several of the ideas from these existing monitors. The specific monitors examined include:

Univac SIP (2)
VAX VMS Display Utility (3)
Burroughs SPARK (4)
IBM RMF/SMF (5)
Tandem XRAY (6)

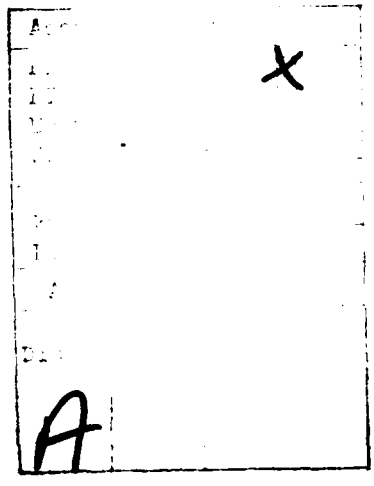
Technical Report TR-952
AFOSR-78-3654
CSC

October 1980

# Design of the VAX Instrumentation Package (VIP)

by

L.W. Dowdy
W.S. Freeze
B.G. Labaw
J.N. Reed



This research was supported in part by the Air Force Office of Scientific Research under Grant AFOSR-78-3654 and also in part by the Computer Center of the University of Maryland.

# Design of the VAX Instrumentation Package (VIP)

L.W. Dowdy, W.S. Freeze, B.G. Labaw, and J.N. Reed

## 1. Introduction

This report describes the design of the VAX Instrumentation Package (VIP). The target machine initially is the VAX 11/780 running under the Unix operating system at the Computer Science Department of the University of Maryland.

VIP is a software monitor of VAX performance. It is designed as an aid for system monitoring, system tuning, and system modeling. For example, VIP monitors the request rate of system files. If the request rate of a particular file becomes high, the system may be "tuned" by making the file resident in main memory. Specific parameters of the file (e.g., the average number of words transferred per file request) may be used in a queuing network model of the system to predict the performance impact of making the file resident in main memory.

A major motivation for VIP is collecting parametric values for, and providing validation of, system models of the VAX. In the past few years, sophisticated and powerful modeling techniques have been developed [1]. A major problem in the application of these techniques is that actual systems do not monitor those input parameters that are necessary for the application of the modeling techniques. Furthermore, these modeling techniques typically provide performance measures which are more extensive than are usually measured. Validation, as well as the construction, of accurate models is thus a difficult problem.

During the design process of VIP, several existing software monitors were examined in detail. VIP includes, extends, and deletes several of the ideas from these existing monitors. The specific monitors examined include:

Univac SIP [2]

VAX VMS Display Utility [3]

Burroughs SPARK [4]

IBM RMF/SMF [5]

Tandem XRAY [6]

The structure of VIP is illustrated in figure 1:

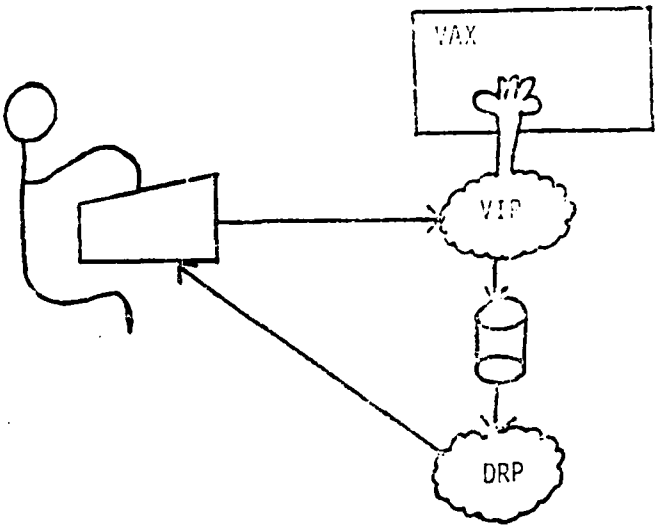



Figure 1: VIP Structure

A user initiates VIP. The user can specify various options. These options include: a) the specific performance information desired, b) the specific process type(s) (e.g., batch, system overhead, Fortran) that the user desires to monitor, and c) timing information describing the resolution and the length of the monitoring session. For example, a user may request cpu and disk utilizations (i.e., option a), for small batch jobs (i.e., option b), to be reported over a 30 minute period in 5 minute intervals (i.e., option c). VIP monitors the requested activity and, according to user specified timing resolutions, logs the information on secondary storage. Concurrently, or at some later time, the logged information is used by a data reduction program, DRP, which analyzes the data and produces reports for the user. Upon receiving initial reports, the user can dynamically alter the parameters used by VIP. Since the timing input parameters and/or the monitored events can be changed, the overhead of running VIP on the system is variable.

Section 2 of this report specifies the data structures used by VIP in recording the measurement information. Section 3 gives the specific

measured entities.

## 2. Data Structures

### 2.1 Record Types

To record information, VIP accesses one of four types of data structures. The data structures associated with a measurable entity depend on its nature and the desired level of information to be acquired.

The four types of data structures are (see Figure 2): .

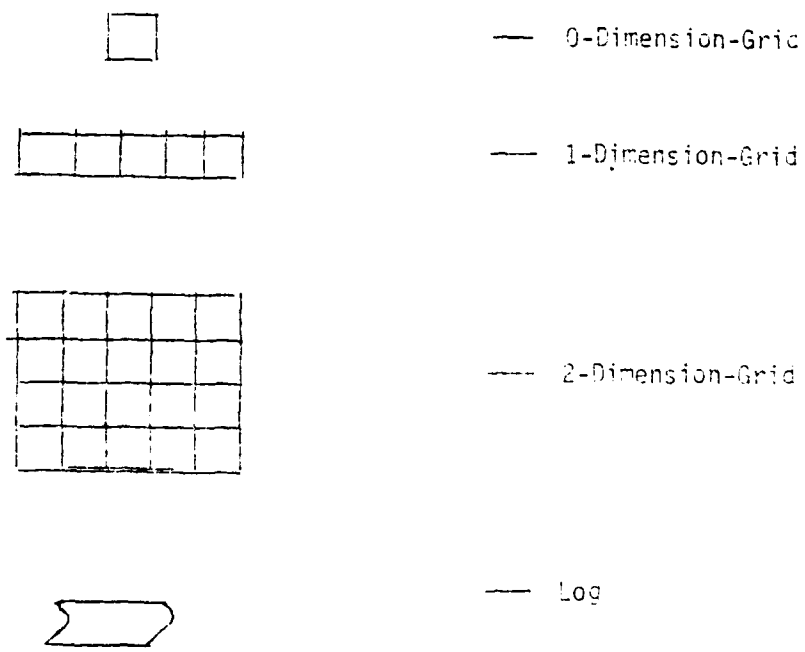



Figure 2: Data Structure Types

A. 0-Dimension-Grid: A 0-dimension-grid record is a single counter. It holds a count of the number of occurences of an

event (e.g., number of page faults).

B. 1-Dimension-Grid: A 1-dimension-grid record is an array of single counters. The particular counter into which an event is recorded depends upon the specific measured quantity. For example, consider the number of words transferred per I/O request to a disk. The number of I/O requests transferring, say 1 to 100 words might be recorded in 1-dimension-grid[1]; the number of I/O requests transferring 101 to 200 words might be recorded in 1-dimension-grid[2]; and so on. In this way the distribution of the event is recorded. Device service time distributions and queue length distributions can be obtained using this data structure. The number of counters and the ranges associated with each counter are user definable.

C. 2-Dimension-Grid: A 2-dimension-grid record is an extension of the 1-dimension-grid record. The user can specify the meaning of each dimension's range. For example, suppose load dependent service time distributions are desired of a disk channel. That is, the average time it takes to satisfy a disk request (i.e., service time) depends upon the number of requests in the disk channel queue. This dependency may be measured. When the queue has more than 1 request present, the average service time per request may be lower since seek activity on different disk units may overlap. The user may specify the first dimension range to be associated with the channel queue length and the second dimension range to be associated with the specific measured quantity. 2-dimension-grid[1,1] might record the number of service requests which require 0 to 5 msec of service when 1 to 3 requests were in the queue; 2-dimension-grid[1,2] might record the number of service requests which require 5 to 10 msec of service when 1 to 3 requests were in the queue; 2-dimension-grid[2,1] might record the number of service requests which require 0 to 5 msec of service when 4 to 6 requests were in the queue; 2-dimension-grid[2,2] might record

the number of service requests which require 5 to 10 msec of service when 4 to 6 requests were in the queue; and so on (see Figure 3). In this way the relative dependence of one measurable quantity parameter of an event to another measurable quantity parameter is obtained.

D. Log - A log record is used when none of the other data structures is appropriate to record the occurence of an event. This is the case when the number of parameters measured at the time of an event is too great to be recorded in the other data structures. A log record is identified with a single occurrence of an event. Specific errors in the system may be recorded in this way.

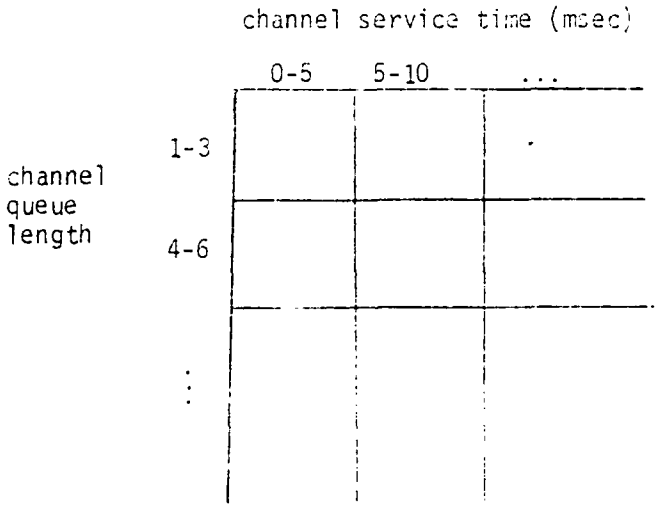



Figure 3: 2-Dimension-Grid Example

## 2.2 Special Job Class Monitoring

A value of 1 or 0, respectively, is assigned to a process to indicate whether or not the process is singled out for special monitoring. VIP records the measurement according to this process type. For example, the count of the number of disk operations might be broken into two parts: the first corresponding to processes associated with

student jobs and the second to all other processes. Processes which might be singled out for special monitoring include: batch job processes, processes associated with jobs having a certain account number range, processes using a certain piece of software. User selection of singled out processes may be done dynamically.

## 2.3 Record Descriptions

### 2.3.1 0-Dimension-Grid

A 0-dimension-grid record contains the number of times an event occurred and information about the event itself.

Format:

```
Type
    0-dimension-grid = record
        idnumber  : integer;
        length    : integer;
        logtime   : integer;
        0-grid    : array[0..1] of integer;
        cumtot    : array[0..1] of integer;
        starttime : array[0..1] of integer;
        end;
```

Explanation:

idnumber - Event identifier.

length - The record length in words.

logtime - The time the record was last logged to secondary storage and reinitialized.

0-grid - 0-grid[i] holds the number of times the event occurred by processes with type i for i = 0 or 1.

cumtot - cumtot[i] holds the cumulative total of the measurement

values (i.e., the actual quantities measured by VIP) associated with processes of type i for i = 0 or 1. For example, to determine the amount of time that a channel is idle while a connected device is busy (e.g., seeking), cumtot[1] would hold this cumulative time for type 1 processes, while cumtot[0] holds the same for all other processes. In contrast, 0-grid[i] would contain the number of times that the channel is idle while the device is busy for type i processes.

starttime - starttime[i] holds the last time VIP accessed the record by a type i process.

Example:

A 0-dimension-grid record might be used to record disk busy times. VIP accesses the record whenever a disk request is initiated or terminated. For an initiation, the starttime is simply recorded. For a termination, the recorded starttime is subtracted from the current time to give the busy time. This number is then added to cumtot, and 0-grid is incremented by one. The average busy time per request can later be calculated by the DRP by dividing cumtot by 0-grid.

### 2.3.2 1-Dimension-Grid

A 1-dimension-grid record is an extension of a 0-dimension-grid record used for capturing information needed to determine distributions. The 1-dimenson-grid record contains several counters, each associated with a range (user definable). A counter is incremented for any event occurrence having a measurement value falling in its given range.

Format:

```
1-dimension-grid = record
    idnumber  : integer;
    length    : integer;
    logtime   : integer;
```

```
size      :integer;
rangeid   :integer;
range     :array[0..size-1] of integer;
1-grid    :array[0..1, 0..size] of integer;
cumtot    :array[0..1] of integer;
starttime :array[0..1] of integer;
end;
```

Explanation:

idnumber - Event identifier.

length - The record length in words.

logtime - The time the record was last logged to secondary storage and reinitialized.

size - The number of counters in the record.

rangeid - Identifying number for the interpretation of the range.

range - Array containing the boundary limits associated with the counters.

1-grid - An array of counters. The first subscript refers to the process type. For example, 1-grid[0,4] counts, for processes with type 0, the event occurrences having measurement values specified by the range boundaries given in range[3] and range[4].

cumtot - cumtot[i] holds the cumulative total of the measurement values associated with processes of type i for i = 0 or 1.

starttime - starttime[i] holds the last time VIP accessed the record of a type i process.

Example:

A 1-dimension-grid record might be used to record information ultimately used to compute the average and distribution of CPU burst times. The rangeid would specify that the range values correspond to

CPU burst times. The array range might contain:

range = [3 5 7 9 ... ]

VIP accesses the record upon each CPU context switch. The CPU burst time is calculated by subtracting starttime from the current time. Suppose batch processes have been singled out for special monitoring (i.e., type 1). For a batch process having a measured burst time of 8 ms, the number 8 is added to cumtot[1] (1 for a batch process type), and 1-grid[1,3] is incremented by one since range[2] < 8 ≤ range[3]. The average batch process burst time can later be calculated by the DRP by dividing cumtot[1] by the total count which is found by summing 1-grid[1,k] for k = 0..size.

### 2.3.3 2-Dimension-Grid

A 2-dimension-grid record is an extension of a 1-dimension-grid record used for capturing information needed to determine distributions where the distribution depends upon some network parameter. The record contains several counters. When an event occurs the specific counter which is updated depends upon: 1) the process type associated with the event and 2) the values of the two parameters specified by rangeid (user definable).

Format:

```
2-dimension-grid = record
    idnumber  :integer;
    length    :integer;
    logtime   :integer;
    size      :array[0..1] of integer;
    rangeid   :array[0..1] of integer;
    range0    :array[0..size[0]-1] of integer;
    range1    :array[0..size[1]-1] of integer;
    2-grid    :array[0..1, 0..size[0], 0..size[1]] of integer;
    cumtot    :array[0..1, 0..size[0]] of integer;
    starttime :array[0..1] of integer;
    end;
```

Explanation:

idnumber - Event identifier.

length - The record length in words.

logtime - The time the record was last logged to secondary storage and reinitialized.

size - Array containing the number of counters in the record in each dimension.

rangeid - Array containing identifying numbers for the interpretation of the range in each dimension.

range0 and range1 - Arrays containing the boundary limits associated with the counters in each dimension.

2-grid - An array of counters. The first subscript refers to the process type. The remaining subscripts refer to respective rangeid interpretations. For example, 2-grid[i,j,k] counts, for processes of type i, the event occurences having measurement values specified by: 1) the range boundaries given in range0[j-1] and range0[j] for the first dimension, and 2) the range boundaries given in range1[k-1] and range1[k] for the second dimension.

cumtot - Array of cumulative totals. The first subscript refers to the process type. The second subscript refers to the first dimension range. For example, cumtot[i,j] holds the cumulative total of measurement values associated with processes of type i whose first dimension interpretation is bounded by range0[j-1] and range0[j].

starttime - starttime[i] holds the last time VIP accessed the record of a type i process.

Example:

Suppose the interarrival time distribution of requests to the swapping disk is desired. However, it might be suspected that the distribution depends upon the number of processes resident in main memory. A 2-dimension-grid record could be used as follows.

```
size = [3  4]
rangeid[0] = "number of processes in main memory"
rangeid[1] = "interarrival times to the swapping disk"
range0 = [3  6  9]
range1 = [5  10  15  20]
```

Suppose a type 0 process makes a request for the disk and this 2-dimension-grid record is accessed. The value of starttime[0] in the record is subtracted from the current time and this difference is taken as the measurement value. The value of starttime[0] is updated to the current time for the next access. Suppose that a measurement value of 22 msec is observed. Suppose that the number of processes in main memory is 8. The counter, 2-grid[0,2,4], would be incremented, since 0 is the process type; range0[1] $<$ 8 $\leq$ range0[2]; and, range1[4] $<$ 22. The measurement value, 22, would be added to cumtot[0,2], since 0 is the process type and range0[1] $<$ 8 $\leq$ range0[2].

The average interarrival time of type i processes to the swapping disk when k processes are resident in main memory can be calculated by the data reduction program. This average time is found by dividing cumtot[0,x] by the sum of 2-grid[0,x,y], where range0[x-1] $<$ k $\leq$ range0[x], and y = 0,1,2,3,4.

### 2.3.4 Log

A log record is specific to a certain event and is created by VIP and then saved.

Format:

```
Type log = record
                idnumber : integer;
                length   : integer;
                logtime  : integer;
                proctype : integer;
                rest     : otherinfo;
                end;
```

Explanation:

idnumber - Event identifier.

length - The record length in words.

logtime - The time the record was logged to secondary storage.

proctype - Holds a value of 1 or 0 indicating the process type of the process causing the event.

rest - (Whatever event specific information that is necessary).

Example:

A log record might be used to gather information about individual jobs in the system. The record would contain fields for the creator and creation time of the job.

## 3. Instrumentation Entities

This section details the individual items to be measured. The type of data structure format used to record the information (described in detail in the previous section) is given for each item. Certain measurements can be derived from other more basic measurements (e.g., utilization = active time/total time). These items are to be obtained from the data reduction program, DRP, and are so indicated.

The individual items are classified into: a) I/O devices, b) I/O controllers, c) terminal interfaces, d) bus resources, e) CPU resources,

f) memory resources, and g) software resources.

| item | data structure |
|---|---|
| a) I/O devices (e.g., RP06 disks, TE16 tapes, DR11 parallel interfaces) | |
| 1) number of words transferred in and out | 1-grid |
| 2) number of calls (i.e., transfers, transactions) | DRP |
| 3) throughput (i.e., calls per unit time) | DRP |
| 4) busy (i.e., active) time | 0-grid |
| 5) utilization | DRP |
| 6) seek time | 1-grid |
| 7) rotational latency time | 1-grid |
| 8) transfer time | 1-grid |
| 9) queue length | 1-grid |
| 10) response time per request (i.e., queue time + service time) | 2-grid |
| 11) blocked (i.e., device idle but queue is nonempty) | 2-grid |
| 12) interarrival time | 2-grid |
| 13) interdeparture time | 2-grid |
| b) I/O controllers (i.e., when more than 1 device has a common controller) | |
| 1) number of words transferred | DRP |
| 2) number of calls | DRP |
| 3) throughput | DRP |
| 4) busy time | 0-grid |
| 5) utilization | DRP |
| 6) queue length | 1-grid |
| 7) response time | 2-grid |
| 8) controller free while device busy | 0-grid |
| 9) controller busy while device free | 0-grid |
| 10) device overlap | 2-grid |
| 11) blocked (i.e., controller idle but queue is nonempty) | 2-grid |
| 12) interarrival time | 2-grid |
| 13) interdeparture time | 2-grid |
| c) terminal interfaces (e.g., DZ11 multiplexors) | |
| 1) number of words transferred | 1-grid |
| 2) number of calls (i.e., process wakeups) | DRP |
| 3) throughput | DRP |
| 4) busy time | 0-grid |
| 5) utilization | DRP |
| 6) queue length | 1-grid |
| 7) think time from terminals | 1-grid |
| 8) response time from system | 1-grid |
| 9) number of active terminals | 1-grid |
| 10) blocked | 2-grid |
| 11) interarrival time | 2-grid |
| 12) interdeparture time | 2-grid |

d) bus resources (e.g., UNIBUS, MASSBUS, SBI, console)

| | | |
|---|---|---|
| 1) | number of words transferred | 1-grid |
| 2) | number of calls | DRP |
| 3) | throughput | DRP |
| 4) | busy time | 0-grid |
| 5) | utilization | DRP |
| 6) | queue length | 1-grid |
| 7) | response time | 2-grid |
| 8) | bus free while unit busy | 0-grid |
| 9) | bus busy while unit free | 0-grid |
| 10) | blocked | 2-grid |
| 11) | interarrival time | 2-grid |
| 12) | interdeparture time | 2-grid |

e) CPU resources (by processor mode)

| | | |
|---|---|---|
| 1) | burst time distribution | 1-grid |
| 2) | number of bursts | DRP |
| 3) | throughput | DRP |
| 4) | utilization | DRP |
| 5) | queue length | 1-grid |
| 6) | response time | 2-grid |
| 7) | number of interrupts (by type) | 0-grid |
| 8) | overhead burst time distribution | 1-grid |
| 9) | overhead number of bursts | DRP |
| 10) | overhead utilization | DRP |
| 11) | monitor overhead time | 0-grid |
| 12) | monitor overhead utilization | DRP |
| 13) | blocked | 2-grid |
| 14) | interarrival time | 2-grid |
| 15) | interdeparture time | 2-grid |

f) memory resources

| | | |
|---|---|---|
| 1) | multiprogramming level (i.e., number of active PCBs) | 1-grid |
| 2) | amount of utilized core | 1-grid |
| 3) | amount of available core | DRP |
| 4) | active time of memory box 0 | 0-grid |
| 5) | active time of memory box 1 | 0-grid |
| 6) | active time of both memory boxes simultaneously | 0-grid |
| 7) | utilization of memory box 0 | DRP |
| 8) | utilization of memory box 1 | DRP |
| 9) | utilization of both memory boxes simultaneously | DRP |
| 10) | working set size | 1-grid |
| 11) | free page list size | 1-grid |
| 12) | page pool size | 1-grid |
| 13) | number of dirty pages (i.e., modify list size) | 0-grid |
| 14) | number of page faults | 0-grid |
| 15) | program size | 1-grid |
| 16) | page cache hit ratio | 0-grid |
| 17) | swap size distribution | 1-grid |
| 18) | number of swaps | DRP |
| 19) | percentage of program residency | 1-grid |
| 20) | page lifetime | 1-grid |
| 21) | number of page reads | 0-grid |
| 22) | number of page writes | 0-grid |
| 23) | interarrival time | 2-grid |
| 24) | interdeparture time | 2-grid |

g) software resources (i.e., any queue for software) (e.g., allocators, schedulers, I/O files, loaders, exception handlers, OS services)

| | |
|---|---|
| 1) number of calls | 0-grid |
| 2) throughput | DRP |
| 3) busy time | 0-grid |
| 4) utilization | DRP |
| 5) queue length | 1-grid |
| 6) response time | 2-grid |
| 7) blocked (i.e., queue nonempty but software resource is idle) | 2-grid |
| 8) interarrival time | 2-grid |
| 9) interdeparture time | 2-grid |

## 4. Summary

A design of a software instrumentation package has been described. The target machine initially is a VAX 11/780. However, the design is applicable to paging/swapping systems in general. A data structure for the collection of information is given, and the information to collect is specified.

## References

[1] Computing Surveys, Special Issue: Queueing Network Models of Computer System Performance, Vol. 10, 3, September 1978.

[2] SIP Level 4R2, Sperry-Univac Software Release Document 138, December 1975.

[3] VAX/VMS System Manager's Guide, DEC Order No. AA-D027A-TE, August 1978.

[4] Systems Performance Analysis Review Kit, Burroughs Corporation B7700/6700 User's Manuel, 1976.

[5] OS/VS2 MVS System Programming Library; System Management Facilities (SMF), IBM Document GC28-0706-1, June 1977.

[6] XRAY Users Manuel Update A03, Tandem Computers Incorporated, January 1980.


DATE
ILME